**INTERVIEW QUESTION**

**ON**

**C++**

**Q.1 INTRODUCTION of C++:**

[**C++**](https://www.interviewbit.com/blog/cpp-ides/) is a powerful and all-purpose programming tool developed by **Bjarne Stroustrup at Bell Labs**. This language is an extension of C and is by far one of the fastest object-oriented programming languages. C++ is super popular because of its high speed and compatibility.

It is widely used in the development of games and servers while some of the real-world applications of C++ are as follows

* Operating systems
* GUI based applications
* Distributed systems
* Database software
* Banking applications
* Advanced computations and graphics
* Embedded systems

**Q.2 What are the various OOPs concepts in C++?**

* Classes: It is a user-defined datatype
* Objects: It is an instance of a class
* Abstraction: It is a technique of showing only necessary details
* Encapsulation: Wrapping of data in a single unit
* Inheritance: The capability of a class to derive properties and characteristics from another class
* Polymorphism: Polymorphism is known as many forms of the same thing

**Q.3. What are the different data types present in C++?**

The 4 data types in C++ are given below:

* **Primitive Datatype(basic datatype).** Example- char, short, int, float, long, double, bool, etc.
* **Derived datatype.** Example- array, pointer, etc.
* **Enumeration**. Example- enum
* **User-defined data types**. Example- structure, class, etc.

**Q.4. What is the difference between C and C++?**

* The main difference between C and C++ are provided in the table below:

| **C** | **C++** |
| --- | --- |
| C is a procedure-oriented programming language. | C++ is an object-oriented programming language. |
| C does not support data hiding. | Data is hidden by encapsulation to ensure that data structures and operators are used as intended. |
| C is a subset of C++ | C++ is a superset of C. |
| Function and operator overloading are not supported in C | Function and operator overloading is supported in C++ |
| Namespace features are not present in C | Namespace is used by C++, which avoids name collisions. |
| Functions can not be defined inside structures. | Functions can be defined inside structures. |
| calloc() and malloc() functions are used for memory allocation and free() function is used for memory deallocation. | new operator is used for memory allocation and deletes operator is used for memory deallocation. |

**Q.5 Define token in C++**

A token is the smallest individual element of a program that is understood by a compiler. A token comprises the following:

Keywords – That contain a special meaning to the compiler

Identifiers – That hold a unique value/identity

Constants – That never change their value throughout the program

Strings – That contains the homogenous sequence of data

Special Symbols – They have some special meaning and cannot be used for another purpose; eg: [] () {}, ; \* = #

Operators – Who perform operations on the operand

**Q.6 Discuss the difference between prefix and postfix?**

| **prefix** | **postfix** |
| --- | --- |
| It simply means putting the operator before the operand | It simply means putting the operator after the operand |
| It executes itself before **‘; ‘** | It executes itself after **‘; ‘** |
| Associativity of prefix ++ is right to left | Associativity of postfix ++ is left to right |

### Q.7. What are class and object in C++?

A class is a user-defined data type that has data members and member functions. Data members are the data variables and member functions are the functions that are used to perform operations on these variables.

An object is an instance of a class. Since a class is a user-defined data type so an object can also be called a variable of that data type.

**Q.8. What is the difference between struct and class?**

In C++ a structure is the same as a class except for a few differences like security. The difference between struct and class are given below:

| **Structure** | **Class** |
| --- | --- |
| Members of the structure are public by default. | Members of the class are private by default. |
| When deriving a struct from a class/struct, default access specifiers for base class/struct are public. | When deriving a class, default access specifiers are private. |

### Q.9 What is operator overloading?

Operator Overloading is a very essential element to perform the operations on user-defined data types. By operator overloading we can modify the default meaning to the operators like +, -, \*, /, <=, etc.

**Q.9 What are the C++ access modifiers?**

The access restriction specified to the class members( whether it is member function or data member) is known as access modifiers/specifiers.

**Access Modifiers are of 3 types:**

1. Private – It can neither be accessed nor be viewed from outside the class
2. Protected – It can be accessed if and only if the accessor is the derived class
3. Public – It can be accessed or be viewed from outside the class

**Q,10 What is polymorphism in C++?**

Polymorphism in simple means having many forms. Its behavior is different in different situations. And this occurs when we have multiple classes that are related to each other by inheritance.

For example, think of a base class called a car that has a method called car brand(). Derived classes of cars could be Mercedes, BMW, Audi - And they also have their own implementation of a cars

The two types of polymorphism in c++ are:

* Compile Time Polymorphism
* Runtime Polymorphism

### Q.11 Explain constructor in C++

The constructor is a member function that is executed automatically whenever an object is created. Constructors have the same name as the class of which they are members so that compiler knows that the member function is a constructor. And no return type is used for constructors.

### Q.12 What is virtual function

**Virtual function** is a member function in the base class that you redefine in a derived class. A virtual function is declared using the virtual keyword. When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer

**Q.13 Compare compile time polymorphism and Runtime polymorphism**

The main difference between compile-time and runtime polymorphism is provided below:

| **Compile-time polymorphism** | **Run time polymorphism** |
| --- | --- |
| In this method, we would come to know at compile time which method will be called. And the call is resolved by the compiler. | In this method, we come to know at run time which method will be called. The call is not resolved by the compiler. |
| It provides fast execution because it is known at the compile time. | It provides slow execution compared to compile-time polymorphism because it is known at the run time. |
| It is achieved by function overloading and operator overloading. | It can be achieved by virtual functions and pointers. |
| Example -  **int** **add**(**int** a, **int** b){  **return** a+b;  }  **int** **add**(**int** a, **int** b, **int** c){  **return** a+b+c;  }  **int** **main**(){  cout<<add(2,3)<<endl;  cout<<add(2,3,4)<<endl;  **return** 0;  } | Example -  **class** **A**{  **public**:  **virtual** **void** **fun**(){  cout<<"base ";  }  };  **class** **B**: **public** A{  **public**:  **void** **fun**(){  cout<<"derived ";  }  };  **int** **main**(){  A \*a=**new** B;  a->fun();  **return** 0;  } |

### Q.14 What do you know about friend class and friend function?

A friend class can access private, protected, and public members of other classes in which it is declared as friends.

Like friend class, friend function can also access private, protected, and public members. But, Friend functions are not member functions.

### Q.15 What are the C++ access specifiers?

In C++ there are the following access specifiers:

**Public:** All data members and member functions are accessible outside the class.

**Protected:** All data members and member functions are accessible inside the class and to the derived class.

**Private:** All data members and member functions are not accessible outside the class.

### Q.16 Define inline function

If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. One of the important advantages of using an inline function is that it eliminates the function calling overhead of a traditional function

### Q.17. What is a reference in C++?

A reference is like a pointer. It is another name of an already existing variable. Once a reference name is initialized with a variable, that variable can be accessed by the variable name or reference name both.

For example-

**int** x=10;

**int** &ref=x; //reference variable

If we change the value of ref it will be reflected in x. Once a reference variable is initialized it cannot refer to any other variable. We can declare an array of pointers but an array of references is not possible.

### Q.18 What do you mean by abstraction in C++?

Abstraction is the process of showing the essential details to the user and hiding the details which we don’t want to show to the user or hiding the details which are irrelevant to a particular user.

### Q.19 Is deconstructor overloading possible? If yes then explain and if no then why?

No destructor overloading is not possible. Destructors take no arguments, so there’s only one way to destroy an object. That’s the reason destructor overloading is not possible.

### Q.20. What do you mean by call by value and call by reference?

In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function.

In call by reference method, we pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.

### Q.21 What is an abstract class and when do you use it?

A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

### Q.22 What are destructors in C++?

A constructor is automatically called when an object is first created. Similarly when an object is destroyed a function called destructor automatically gets called. A destructor has the same name as the constructor (which is the same as the class name) but is preceded by a tilde.

### Q.23 What are the static members and static member functions?

When a variable in a class is declared static, space for it is allocated for the lifetime of the program. No matter how many objects of that class have been created, there is only one copy of the static member. So same static member can be accessed by all the objects of that class.

A static member function can be called even if no objects of the class exist and the static function are accessed using only the class name and the scope resolution operator ::

### Q.24 Explain inheritance

Inheritance is the process of creating new classes, called derived classes, from existing classes. These existing classes are called base classes. The derived classes inherit all the capabilities of the base class but can add new features and refinements of their own.

### Q.25 Explain the constructor in C++

A constructor is a special type of member function of a class, whose name is the same as that of the class by whom it is invoked and initializes value to the object of a class.

There are 3 types of constructors:

1. **Default constructor:**It is the most basic type of constructor which accepts no arguments or parameters. Even if it is not called the compiler calls it automatically when an object is created.
2. **Parameterized constructor:**It is a type of constructor which accepts arguments or parameters. It has to be called explicitly by passing values in the arguments as these arguments help initialize an object when it is created. It also has the same name as that of the class. Also, It is used to overload constructors.

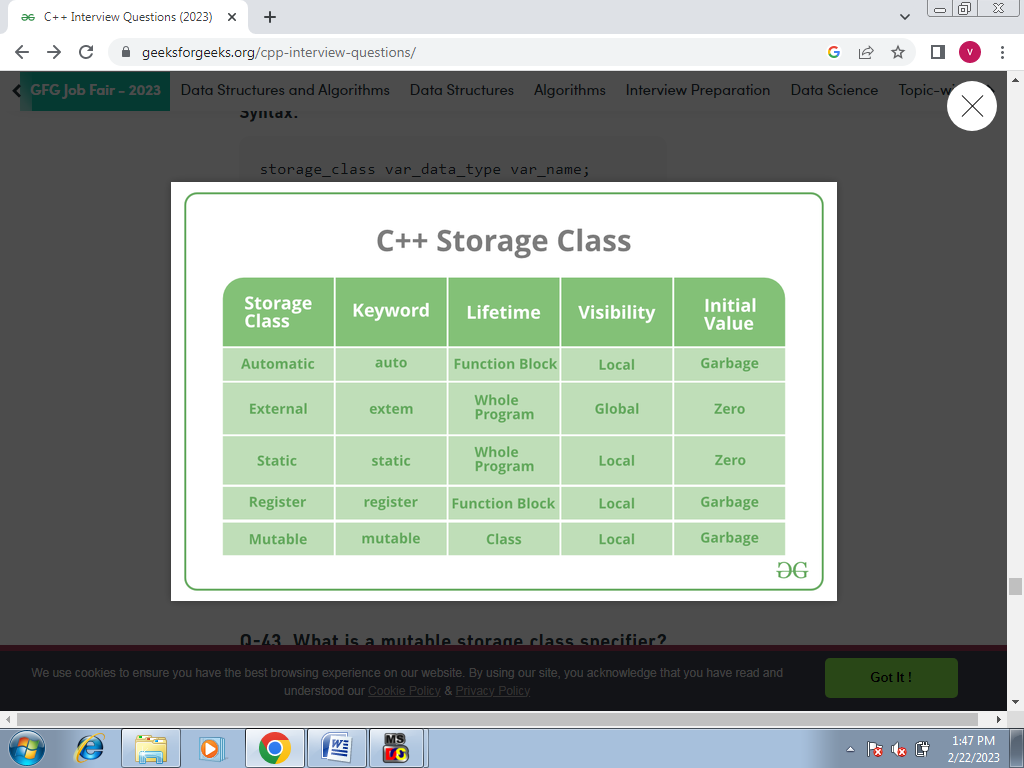
**C. Copy Constructor:**A copy constructor is a member function that initializes an object using another object of the same class. Also, the Copy constructor takes a reference to an object of the same class as an argument.

### Q.26 Define inline function. Can we have a recursive inline function in C++?

An inline function is a form of request not an order to a compiler which results in the inlining of our function to the main function body. An inline function can become overhead if the execution time of the function is less than the switching time from the caller function to called function. To make a function inline use the keyword **inline** before and define the function before any calls are made to the function.

### Q.27 Define storage class in C++ and name some

Storage class is used to define the features(lifetime and visibility) of a variable or function. These features usually help in tracing the existence of a variable during the runtime of a program.



### Q. 28 What does the Scope Resolution operator do?

A scope resolution operator is denoted by a ‘::‘ symbol. Just like its name this operator resolves the barrier of scope in a program. A scope resolution operator is used to reference a member function or a global variable out of their scope furthermore to which it can also access the concealed variable or function in a program.

Scope Resolution is used for numerous amounts of tasks:

* To access a global variable when there is a local variable with the same name
* To define the function outside the class
* In case of multiple inheritances
* For namespace

### Q.29 What is the difference between shallow copy and deep copy?

The difference between shallow copy and a deep copy is given below:

| **Shallow Copy** | **Deep Copy** |
| --- | --- |
| Shallow copy stores the references of objects to the original memory address. | Deep copy makes a new and separate copy of an entire object with its unique memory address. |
| Shallow copy is faster. | Deep copy is comparatively slower. |
| Shallow copy reflects changes made to the new/copied object in the original object. | Deep copy doesn’t reflect changes made to the new/copied object in the original object |

### Q.30 What is the difference between virtual functions and pure virtual functions?

A virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword.

A pure virtual function is a function that has no implementation and is declared by assigning 0. It has no body.

### Q.31 If class D is derived from a base class B. When creating an object of type D in what order would the constructors of these classes get called?

The derived class has two parts, a base part, and a derived part.  When C++ constructs derived objects, it does so in phases. First, the most-base class(at the top of the inheritance tree) is constructed. Then each child class is constructed in order until the most-child class is constructed last.   
So the first Constructor of class B will be called and then the constructor of class D will be called.

During the destruction exactly reverse order is followed. That is destructor starts at the most-derived class and works its way down to base class.  
So the first destructor of class D will be called and then the destructor of class B will be called.

### Q.32 Can we call a virtual function from a constructor?

Yes, we can call a virtual function from a constructor. But the behavior is a little different in this case. When a virtual function is called, the virtual call is resolved at runtime. It is always the member function of the current class that gets called. That is the virtual machine doesn’t work within the constructor.

### Q.33 What are void pointers?

A void pointer is a pointer which is having no datatype associated with it. It can hold addresses of any type.

For example-

**void** \*ptr;

**char** \*str;

p=str; // no error

str=p; // error because of type mismatch

We can assign a pointer of any type to a void pointer but the reverse is not true unless you typecast it as

str=(**char**\*) ptr;

### Q.34 What is this pointer in C++?

The member functions of every object have a pointer named this, which points to the object itself. The value of this is set to the address of the object for which it is called. It can be used to access the data in the object it points to.

### Q.35 How do you allocate and deallocate memory in C++?

The new operator is used for memory allocation and deletes operator is used for memory deallocation in C++.

**For example-**

**int** value=**new** **int**; //allocates memory for storing 1 integer

**delete** value; // deallocates memory taken by value

**int** \*arr=**new** **int**[10]; //allocates memory for storing 10 int

**delete** []arr; // deallocates memory occupied by arr

**Q.36 What is an Exception?**

Conditions responsible for creating errors during the execution of a program are known as Exceptions. These errors can interrupt the execution of the program and if the program can’t handle these exceptions then OS handles them and the program is terminated abruptly.

**Q.37 Explain Synchronous Exceptions and Asynchronous Exceptions?**

Synchronous exceptions are the exceptions that occur at a particular instruction. They can only be originated from throw expressions and are caused due to errors like incorrect input or array out-of-index access in a program.  
Asynchronous exceptions are the exceptions that create errors that are not controllable by the program. For example-hardware malfunctions, disk failure, etc.

**Q.38 What is Exception Handling?**

Conditions responsible for creating errors during the execution of a program are known as Exceptions. Handling these exceptions by either removing these conditions or by using some other operations than normal operations is known as exception handling.  Exception handling is an effective means to handle the runtime errors that disrupt the normal flow of the program.

**Q.39 Why do we need exception Handling?**

We use Exception Handling for the following reasons:

1. Separate Error code from Normal code to help us understand errors easily.
2. Functions/Methods can be handled only by the exceptions they choose. The exceptions not chosen will be handled by the caller.
3. Exceptional handling allows the grouping of error types which helps in categorizing them.
4. It makes the program’s error type easy to understand.

**Q.40 How to implement exception handling in C++?**

C++ supports exception handling. It is implemented by **try{ }** and **catch( ){ }** statements.

* The**try** statement allows you to define a block of code to be tested for errors while it is being executed.
* The**throw** keyword throws an exception when a problem is detected, which lets us create a custom error.
* The **catch** statement allows you to define a block of code to be executed if an error occurs in the try block.

It follows certain rules:

* The catch can be executed in multiple ways according to the argument thrown by the try.
* There can be multiple catch( ) but only a single try.
* If try throws but catch is not able to catch it then terminate() will be called by default.
* If no argument is executed it will just continue to the statement after the catch block.

**Q.41 What will happen if an exception is thrown but not caught anywhere?**

* When an exception is thrown but not caught anywhere, the program will terminate abnormally.

**Q.42 What is file Handling?**

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).

**Q.43 How to achieve the File Handling**

For achieving file handling we need to follow the following steps:-  
 STEP 1-Naming a file  
 STEP 2-Opening a file  
 STEP 3-Writing data into the file  
 STEP 4-Reading data from the file  
 STEP 5-Closing a file.

**Q.44 Classes for File stream operations**

The I/O system of C++ contains a set of classes which define the file handling methods. These include ifstream, ofstream and fstream classes. These classes are derived from fstream and from the corresponding iostream class.

**ofstream:** Stream class to write on files   
**ifstream:** Stream class to read from files   
**fstream:** Stream class to both read and write from/to files.

**Q.45 Default Open Modes :**

|  |  |
| --- | --- |
| ifstream | ios::in |
| ofstream | ios::out |
| fstream | ios::in | ios::out |

---------------------------------\*\*\*\*\*\*\*\*\*\*------------------------------------------------------